2

Network Administration

This chapter provides information useful for network administration of Parallel Computing Toolbox software and MATLAB Distributed Computing Server software.

- "Preparing for Parallel Computing" on page 2-2
- "Installing and Configuring" on page 2-5
- "Using a Different MPI Build on UNIX Operating Systems" on page 2-6
- "Shutting Down a Job Manager Configuration" on page 2-9
- "Customizing Server Services" on page 2-13
- "Accessing Service Record Files" on page 2-17
- "Troubleshooting" on page 2-19

Preparing for Parallel Computing

In this section...

"Before You Start" on page 2-2
"Planning Your Network Layout" on page 2-2
"Network Requirements" on page 2-3
"Fully Qualified Domain Names" on page 2-3
"Security Considerations" on page 2-4

This section discusses the requirements and configurations for your network to support parallel computing.

Before You Start

Before attempting to install Parallel Computing Toolbox software and MATLAB Distributed Computing Server software, read Chapter 1, "Introduction" to familiarize yourself with the concepts and vocabulary of the products.

Planning Your Network Layout

Generally, it is easy to decide which machines will run worker processes and which will run client processes. Worker sessions usually run on the cluster of machines dedicated to that purpose. The MATLAB client session usually runs where MATLAB programs are run, often on a user's desktop.

The job manager process should run on a stable machine, with adequate resources to manage the number of tasks and amount of data expected in your parallel computing applications.

The following table shows what products and processes are needed for each of these roles in the parallel computing configuration.

Session	Product	Processes
Client	Parallel Computing Toolbox	MATLAB with toolbox
Worker	MATLAB Distributed Computing Server	worker; mdce service (if using a job manager)
Job manager	MATLAB Distributed Computing Server	mdce service; job manager

The server software includes the mdce service or daemon. The mdce service is separate from the worker and job manager processes, and it must be running on all machines that run job manager sessions or workers that are registered with a job manager. (The mdce service is not used with third-party schedulers.)

You can install both toolbox and server software on the same machine, so that one machine can run both client and server sessions.

Network Requirements

To view the network requirements for MATLAB Distributed Computing Server software, visit the product requirements page on the MathWorks Web site at

```
http://www.mathworks.com/products/distriben/requirements.html
```

Fully Qualified Domain Names

MATLAB Distributed Computing Server software and Parallel Computing Toolbox software support both short hostnames and fully qualified domain names. The default usage is short hostnames. If your network requires fully qualified hostnames, you can use the `mdce_def` file to identify the worker nodes by their full names. See “Customizing Server Services” on page 2-13. To set the hostname used for a MATLAB client session, see the `pctconfig` reference page.

Security Considerations

The parallel computing products do not provide any security measures. Therefore, be aware of the following security considerations:

- MATLAB workers run as whatever user the administrator starts the node's mdce service under. By default, the mdce service starts as `root` on UNIX operating systems, and as `LocalSystem` on Microsoft Windows operating systems. Because MATLAB provides system calls, users can submit jobs that execute shell commands.
- The mdce service does not enforce any access control or authentication. Anyone with local or remote access to the mdce services can start and stop their workers and job managers, and query for their status.
- The job manager does not restrict access to the cluster, nor to job and task data. Using a third-party scheduler instead of the MathWorks job manager could allow you to take advantage of the security measures it provides.
- The parallel computing processes must all be on the same side of a firewall, or you must take measures to enable them to communicate with each other through the firewall. Workers running tasks of the same parallel job cannot be firewalled off from each other, because their MPI-based communication will not work.
- If certain ports are restricted, you can specify the ports used for parallel computing. See "Defining the Script Defaults" on page 2-13.
- If your network supports multicast, the parallel computing processes accommodate multicast. However, because multicast is disabled on many networks for security reasons, you might require unicast communication between parallel computing processes. Most examples of parallel computing scripts and functions in this documentation show unicast usage.
- If your organization is a member of the Internet Multicast Backbone (MBone), make sure that your parallel computing cluster is isolated from MBone access if you are using multicast for parallel computing. This is generally the default condition. If you have any questions about MBone membership, contact your network administrator.

Installing and Configuring

To find the most up-to-date instructions for installing and configuring the current or past versions of the parallel computing products, visit the MathWorks Web site at

```
http://www.mathworks.com/support/product/DM/installation/ver_current/
```

Using a Different MPI Build on UNIX Operating Systems

In this section...

"Building MPI" on page 2-6

"Using Your MPI Build" on page 2-6

Building MPI

To use an MPI build that differs from the one provided with Parallel Computing Toolbox, this stage outlines the steps for creating an MPI build. If you already have an alternative MPI build, proceed to "Using Your MPI Build" on page 2-6.

1 Unpack the MPI sources into the target file system on your machine. For example, suppose you have downloaded `mpich2-distro.tgz` and want to unpack it into `/opt` for building:

```
# cd /opt
# mkdir mpich2 && cd mpich2
# tar zxvf path/to/mpich2-distro.tgz
# cd mpich2-1.0.8
```

2 Build your MPI using the `enable-sharedlibs` option (this is vital, as you must build a shared library MPI, binary compatible with `MPICH2-1.0.8` for R2009b and later). For example, the following commands build an MPI with the `nemesis` channel device and the `gforker` launcher.

```
# ./configure -prefix=/opt/mpich2/mpich2-1.0.8 \
 --enable-sharedlibs=gcc \
 --with-device=ch3:nemesis \
 --with-pm=gforker 2>&1 | tee log
# make 2>&1 | tee -a log
# make install 2>&1 | tee -a log
```

Using Your MPI Build

When your MPI build is ready, this stage highlights the steps to use it. To get the Parallel Computing Toolbox mpiexec scheduler working with a different MPI build, follow these steps. Most of these steps are also needed if you want to use a different MPI build with third party-schedulers (LSF, generic).

1. Test your build by running the `mpiexec` executable. The build should be ready to test if its `bin/mpiexec` and `lib/libmpich.so` are available in the MPI installation location.

   Following the example in "Building MPI" on page 2-6, `/opt/mpich2/mpich2-1.0.8/bin/mpiexec` and `/opt/mpich2/mpich2-1.0.8/lib/libmpich.so` are ready to use, so you can test the build with:

   ```
   $ /opt/mpich2/mpich2-1.0.8/bin/mpiexec -n 4 hostname
   ```

2. Create an `mpiLibConf` function to direct Parellel Computing Toolbox to use your new MPI. Write your `mpiLibConf.m` to return the appropriate information for your build. For example:

   ```
   function [primary, extras] = mpiLibConf
   primary = '/opt/mpich2/mpich2-1.0.8/lib/libmpich.so';
   extras  = {};
   ```

   The `primary` path *must* be valid *on the cluster*; and your `mpiLibConf.m` file must be higher on the cluster workers' path than *`matlabroot`*`/toolbox/distcomp/mpi`. (Sending `mpiLibConf.m` as a file dependency for this purpose does not work. You can get the `mpiLibConf.m` function on the worker path by either moving the file into a directory on the path, or by having the scheduler use `cd` in its command so that it starts the MATLAB worker from within the directory that contains the function.)

3. Determine necessary daemons and command-line options.

   - Determine all necessary daemons (often something like `mpdboot` or `smpd`). The `gforker` build example in this section uses an MPI that needs no services or daemons running on the cluster, but it can use only the local machine.
   - Determine the correct command-line options to pass to `mpiexec`.

4. Use one of the following options to set up your scheduler to use your new MPI build:

   - For the simplest case of the mpiexec scheduler, set up a configuration to use the `mpiexec` executable from your new MPI build. It is crucial that you use matching `mpiexec`, MPI library, and any daemons (if

any), together. Set the configuration's `MpiexecFileName` property to `/opt/mpich2/mpich2-1.0.8/bin/mpiexec`.

- If you are using a generic scheduler or LSF, modify your parallel wrapper script to pick up the correct `mpiexec`. Additionally, there may be a stage in the wrapper script where the MPI daemons are launched.

  The parallel submission wrapper script must:

  - Determine which nodes are allocated by the scheduler.
  - Start required daemon processes. For example, for the MPD process manager this means calling `"mpdboot -f <nodefile>"`.
  - Define which mpiexec executable to use for starting workers.
  - Stop the daemon processes. For example, for the MPD process manager this means calling `"mpdallexit"`.

  For examples of parallel wrapper scripts, see *`matlabroot`*`/toolbox/distcomp/examples/integration/`; specifically for an example of Sun Grid Engine, look in the folder `sge` for `sgeParallelWrapper.sh`. Adopt and modify the appropriate script for your particular cluster usage.

Shutting Down a Job Manager Configuration

In this section...
"UNIX and Macintosh Operating Systems" on page 2-9
"Microsoft Windows Operating Systems" on page 2-11

If you are done using the job manager and its workers, you might want to shut down the server software processes so that they are not consuming network resources. You do not need to be at the computer running the processes that you are shutting down. You can run these commands from any machine with network access to the processes. The following sections explain shutting down the processes for different platforms.

UNIX and Macintosh Operating Systems

Enter the commands of this section at the prompt in a UNIX shell.

Stopping the Job Manager and Workers

1. To shut down the job manager, enter the commands

   ```
   cd matlabroot/toolbox/distcomp/bin
   ```

   (Enter the following command on a single line.)

   ```
   stopjobmanager -remotehost <job manager hostname> -name
   <MyJobManager> -v
   ```

   If you have more than one job manager running, stop each of them individually by host and name.

   For a list of all options to the script, type

   ```
   stopjobmanager -help
   ```

2. For each MATLAB worker you want to shut down, enter the commands

   ```
   cd matlabroot/toolbox/distcomp/bin
   stopworker -remotehost <worker hostname> -v
   ```

If you have more than one worker session running, you can stop each of them individually by host and name.

```
stopworker -name worker1 -remotehost <worker hostname>
stopworker -name worker2 -remotehost <worker hostname>
```

For a list of all options to the script, type

```
stopworker -help
```

Stopping and Uninstalling the mdce Daemon

Normally, you configure the mdce daemon to start at system boot time and continue running until the machine shuts down. However, if you plan to uninstall the MATLAB Distributed Computing Server product from a machine, you might want to uninstall the mdce daemon also, because you no longer need it.

Note You must have `root` privileges to stop or uninstall the mdce daemon.

---

1. Use the following command to stop the mdce daemon:

   ```
   /etc/init.d/mdce stop
   ```

2. Remove the installed link to prevent the daemon from starting up again at system reboot:

   ```
   cd /etc/init.d/
   rm mdce
   ```

Stopping the Daemon Manually. If you used the alternative manual startup of the mdce daemon, use the following commands to stop it manually:

```
cd matlabroot/toolbox/distcomp/bin
mdce stop
```

Microsoft Windows Operating Systems

Stopping the Job Manager and Workers

Enter the commands of this section at the prompt in a DOS command window.

1. To shut down the job manager, enter the commands

   ```
   cd matlabroot\toolbox\distcomp\bin
   ```

   (Enter the following command on a single line.)

   ```
   stopjobmanager -remotehost <job manager hostname> -name
   <MyJobManager> -v
   ```

   If you have more than one job manager running, stop each of them individually by host and name.

   For a list of all options to the script, type

   ```
   stopjobmanager -help
   ```

2. For each MATLAB worker you want to shut down, enter the commands

   ```
   cd matlabroot\toolbox\distcomp\bin
   stopworker -remotehost <worker hostname> -name <worker name> -v
   ```

   If you have more than one worker session running, you can stop each of them individually by host and name.

   ```
   stopworker -remotehost <worker hostname> -name <worker1 name>
   stopworker -remotehost <worker hostname> -name <worker2 name>
   ```

   For a list of all options to the script, type

   ```
   stopworker -help
   ```

Stopping and Uninstalling the mdce Service

Normally, you configure the mdce service to start at system boot time and continue running until the machine shuts down. If you need to stop the mdce

service while leaving the machine on, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin
mdce stop
```

If you plan to uninstall the MATLAB Distributed Computing Server product from a machine, you might want to uninstall the mdce service also, because you no longer need it.

You do not need to stop the service before uninstalling it.

To uninstall the mdce service, enter the following commands at a DOS command prompt:

```
cd matlabroot\toolbox\distcomp\bin
mdce uninstall
```

Customizing Server Services

In this section...
"Defining the Script Defaults" on page 2-13
"Overriding the Script Defaults" on page 2-15

The MATLAB Distributed Computing Server scripts run using several default parameters. You can customize the scripts, as described in this section.

Defining the Script Defaults

The scripts for the server services require values for several parameters. These parameters set the process name, the user name, log file location, ports, etc. Some of these can be set using flags on the command lines, but the full set of user-configurable parameters are in the `mdce_def` file.

Note The startup script flags take precedence over the settings in the `mdce_def` file.

The default parameters used by the server service scripts are defined in the file:

- *`matlabroot`*`\toolbox\distcomp\bin\mdce_def.bat` (on Microsoft Windows operating systems)
- *`matlabroot`*`/toolbox/distcomp/bin/mdce_def.sh` (on UNIX or Macintosh operating systems)

To set the default parameters, edit this file before installing or starting the mdce service.

The `mdce_def` file is self-documented, and includes explanations of all its parameters.

Note If you want to run more than one job manager on the same machine, they must all have unique names. Specify the names using flags with the startup commands.

Setting the User

By default, the job manager and worker services run as the user who starts them. You can run the services as a different user with the following settings in the `mdce_def` file.

Parameter	Description
`MDCEUSER`	Set this parameter to run the mdce services as a user different from the user who starts the service. On a UNIX operating system, set the value before starting the service; on a Windows operating system, set it before installing the service.
`MDCEPASS`	On a Windows operating system, set this parameter to specify the password for the user identified in the `MDCEUSER` parameter; otherwise, the system prompts you for the password when the service is installed.

On UNIX operating systems, `MDCEUSER` requires that the current machine has the `sudo` utility installed, and that the current user be allowed to use `sudo` to execute commands as the user identified by `MDCEUSER`. For further information, refer to your system documentation on the `sudo` and `sudoers` utilities (for example, `man sudo` and `man sudoers`).

On Windows operating systems, when executing the `mdce start` script, the user defined by `MDCEUSER` must be listed among those who can log on as a service. To see the list of valid users, select the Windows menu **Start > Settings > Control Panel**. Double-click `Administrative Tools`, then `Local Security Policy`. In the tree, select `User Rights Assignment`, then in the right pane, double-click `Log on as a service`. This dialog box must list the user defined for `MDCEUSER` in your `mdce_def.bat` file. If not, you can add the user to this dialog box according to the instructions in the `mdce_def.bat` file, or when running `mdce start`, you can use another `mdce_def.bat` file that specifies a listed user.

Overriding the Script Defaults

Specifying an Alternative Defaults File

The default parameters used by the mdce service, job managers, and workers are defined in the file:

- *matlabroot*\toolbox\distcomp\bin\mdce_def.bat (on Windows operating systems)
- *matlabroot*/toolbox/distcomp/bin/mdce_def.sh (on UNIX or Macintosh operating systems)

Before installing and starting the mdce service, you can edit this file to set the default parameters with values you require.

Alternatively, you can make a copy of this file, modify the copy, and specify that this copy be used for the default parameters.

On UNIX or Macintosh operating systems, enter the command

```
mdce start -mdcedef my_mdce_def.sh
```

On Windows operating systems, enter the command

```
mdce install -mdcedef my_mdce_def.bat
mdce start -mdcedef my_mdce_def.bat
```

If you specify a new mdce_def file instead of the default file for the service on one computer, the new file is not automatically used by the mdce service on other computers. If you want to use the same alternative file for all your mdce services, you must specify it for each mdce service you install or start.

For more information, see "Defining the Script Defaults" on page 2-13.

Note The startup script flags take precedence over the settings in the mdce_def file.

Starting in a Clean State

When a job manager or worker starts up, it normally resumes its session from the past. This way, a job queue is not destroyed or lost if the job manager machine crashes or if the job manager is inadvertently shut down. To start up a job manager or worker from a clean state, with all history deleted, use the `-clean` flag on the `start` command:

```
startjobmanager -clean -name MyJobManager
startworker -clean -jobmanager MyJobManager
```

Accessing Service Record Files

In this section...
"Locating Log Files" on page 2-17 "Locating Checkpoint Directories" on page 2-18

The MATLAB Distributed Computing Server services generate various record files in the normal course of their operations. The mdce service, job manager, and worker sessions all generate such files. This section describes the types of information stored by the services.

Locating Log Files

Log files for each service contain entries for the service's operations. These might be of particular interest to the network administrator in cases when problems arise.

Operating System	File Location
Windows	The default location of the log files is `<TEMP>\MDCE\Log`, where `<TEMP>` is the value of the system `TEMP` variable. For example, if `TEMP` is set to `C:\TEMP`, the log files are placed in `C:\TEMP\MDCE\Log`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mdce_def.bat` file before starting the mdce service.
UNIX and Macintosh	The default location of the log files is `/var/log/mdce/`. You can set alternative locations for the log files by modifying the `LOGBASE` setting in the `mdce_def.sh` file before starting the mdce service.

Locating Checkpoint Directories

Checkpoint directories contain information related to persistence data, which the server services use to create continuity from one instance of a session to another. For example, if you stop and restart a job manager, the new session continues the old session, using all the same data.

A primary feature offered by the checkpoint directories is in crash recovery. This allows server services to automatically resume their sessions after a system goes down and comes back up, minimizing the loss of data. However, if a MATLAB worker goes down during the evaluation of a task, that task is neither reevaluated nor reassigned to another worker. In this case, a finished job may not have a complete set of output data, because data from any unfinished tasks might be missing.

Note If a job manager crashes and restarts, its workers can take up to 2 minutes to reregister with it.

Platform	File Location
Windows	The default location of the checkpoint directories is `<TEMP>\MDCE\Checkpoint`, where `<TEMP>` is the value of the system `TEMP` variable. For example, if `TEMP` is set to `C:\TEMP`, the checkpoint directories are placed in `C:\TEMP\MDCE\Checkpoint`. You can set alternative locations for the checkpoint directories by modifying the `CHECKPOINTBASE` setting in the `mdce_def.bat` file before starting the mdce service.
UNIX and Macintosh	The checkpoint directories are placed by default in `/var/lib/mdce/`. You can set alternative locations for the checkpoint directories by modifying the `CHECKPOINTBASE` setting in the `mdce_def.sh` file before starting the mdce service.

Troubleshooting

In this section...
"License Errors" on page 2-19
"Verifying Multicast Communications" on page 2-21
"Memory Errors on UNIX Operating Systems" on page 2-22
"Running Server Processes from a Windows Network Installation" on page 2-22
"Required Ports" on page 2-23
"Ephemeral TCP Ports with Job Manager" on page 2-24

This section offers advice on solving problems you might encounter with MATLAB Distributed Computing Server software.

License Errors

When starting a MATLAB worker, a licensing problem might result in the message

```
License checkout failed. No such FEATURE exists.
License Manager Error -5
```

There are many reasons why you might receive this error:

- This message usually indicates that you are trying to use a product for which you are not licensed. Look at your `license.dat` file located within your MATLAB installation to see if you are licensed to use this product.
- If you are licensed for this product, this error may be the result of having extra carriage returns or tabs in your license file. To avoid this, ensure that each line begins with either `#`, `SERVER`, `DAEMON`, or `INCREMENT`.

  After fixing your `license.dat` file, restart your license manager and MATLAB should work properly.
- This error may also be the result of an incorrect system date. If your system date is before the date that your license was made, you will get this error.

- If you receive this error when starting a worker with MATLAB Distributed Computing Server software:
  - You may be calling the `startworker` command from an installation that does not have access to a worker license. For example, starting a worker from a client installation of the Parallel Computing Toolbox product causes the following error:

```
The mdce service on the host hostname
returned the following error:

Problem starting the MATLAB worker.

The cause of this problem is:

================================================================
   Most likely, the MATLAB worker failed to start due to a
   licensing problem, or MATLAB crashed during startup.  Check
   the worker log file
   /tmp/mdce_user/node_node_worker_05-11-01_16-52-03_953.log
   for more detailed information.  The mdce log file
   /tmp/mdce_user/mdce-service.log
   may also contain some additional information.
================================================================
```

    In the worker log files, you see the following information:

```
License checkout failed.
License Manager Error -15
MATLAB is unable to connect to the license server.
Check that the license manager has been started, and that the
MATLAB client machine can communicate with the license server.

Troubleshoot this issue by visiting:
http://www.mathworks.com/support/lme/R2009a/15

Diagnostic Information:
Feature: MATLAB_Distrib_Comp_Engine
License path: /apps/matlab/etc/license.dat
FLEXnet Licensing error: -15,570. System Error: 115
```

- If you installed only the Parallel Computing Toolbox product, and you are attempting to run a worker on the same machine, you will receive this error because the MATLAB Distributed Computing Server product is not installed, and therefore the worker cannot obtain a license.

Verifying Multicast Communications

Note Although Version 4 of the parallel computing products continues to support multicast communications between its processes, multicast is not recommended and might not be supported in future releases.

Multicast, unlike TCP/IP or UDP, is a subscription-based protocol where a number of machines on a network indicate to the network their interest in particular packets originating somewhere on that network. By contrast, both UDP and TCP packets are always bound for a single machine, usually indicated by its IP address.

The main tools for investigating this type of packet are:

- `tcpdump` for UNIX operating systems
- `winpcap` and `ethereal` for Microsoft Windows operating systems
- A Java™ class included with Version 3 of the parallel computing products.

The Java class is called `com.mathworks.toolbox.distcomp.test.MulticastTester`. Both its static main method and its constructor take two input arguments: the multicast group to join and the port number to use.

This Java class has a number of simple methods to attempt to join a specified multicast group. Once the class has successfully joined the group, it has methods to send messages to the group, listen for messages from the group, and display what it receives. The class can be used both inside MATLAB and from a call to Java software.

Inside MATLAB, use the class as follows:

```
m = com.mathworks.toolbox.distcomp.test.MulticastTester('239.1.1.1', 9999);
m.startSendingThread;
```

```
m.startListeningThread;
 0 : host1name : 0
 1 : host2name : 0
```

From a shell prompt, type (assuming that `java` is on your path)

```
java -cp distcomp.jar com.mathworks.toolbox.distcomp.test.MulticastTester
0 : host1name : 0
1 : host2name : 0
```

Memory Errors on UNIX Operating Systems

If the number of threads created by the server services on a machine running a UNIX operating system exceeds the limitation set by the `maxproc` value, the services fail and generate an out-of-memory error. Check your `maxproc` value on a UNIX operating system with the `limit` command. (Different versions of UNIX software might have different names for this property instead of `maxproc`, such as `descriptors` on Solaris™ operating systems.)

Running Server Processes from a Windows Network Installation

Many networks are configured not to allow `LocalSystem` to have access to UNC or mapped network shares. In this case, run the mdce process under a different user with rights to log on as a service. See “Setting the User” on page 2-14.

Required Ports

Using a Job Manager

BASE_PORT. The `mdce_def` file specifies and describes the ports required by the job manager and all workers. See the following file in the MATLAB installation used for each cluster process:

- *`matlabroot`*`/toolbox/distcomp/bin/mdce_def.sh` (on UNIX operating systems)
- *`matlabroot`*`\toolbox\distcomp\bin\mdce_def.bat` (on Windows operating systems)

Parallel Jobs. On worker machines running a UNIX operating system, the number of ports required by MPICH for the running of parallel jobs ranges from `BASE_PORT + 1000` to `BASE_PORT + 2000`.

Using a Third-Party Scheduler

Before the worker processes start, you can control the range of ports used by the workers for parallel jobs by defining the environment variable `MPICH_PORT_RANGE` with the value `minport:maxport`.

Client Ports

With the `pctconfig` function, you specify the ports used by the client. If the default ports cannot be used, this function allows you to configure ports separately for communication with the job manager and communication with pmode or a MATLAB pool.

Ephemeral TCP Ports with Job Manager

If you use the job manager on a cluster of nodes running Windows operating systems, you must make sure that a large number of ephemeral TCP ports are available on the job manager machine. By default, the maximum valid ephemeral TCP port number on a Windows operating system is 5000, but transfers of large data sets might fail if this setting is not increased. In particular, if your cluster has 32 or more workers, you should increase the maximum valid ephemeral TCP port number using the following procedure:

1. Start the Registry Editor.
2. Locate the following subkey in the registry, and click **Parameters**:

   ```
   HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\Tcpip\Parameters
   ```

3. On the Registry Editor window, select **Edit > New > DWORD Value**.
4. In the list of entries on the right, change the new value name to `MaxUserPort` and press **Enter**.
5. Right-click on the `MaxUserPort` entry name and select **Modify**.
6. In the Edit DWORD Value dialog, enter `65534` in the **Value data** field. Select `Decimal` for the **Base** value. Click **OK**.

   This parameter controls the maximum port number that is used when a program requests any available user port from the system. Typically, ephemeral (short-lived) ports are allocated between the values of 1024 and 5000 inclusive. This action allows allocation for port numbers up to 65534.
7. Quit the Registry Editor.
8. Reboot your machine.